GLOBAL ROBOTICS CHALLENGE

RULE BOOK 2026

GLOBAL ROBOTICS CHALLENGE

PYTHON DEVELOPMENT

Smart Programmers CHALLENGE

For More Information:

Scan QR Code

GLOBAL ROBOTICS CHALLENGE

1. Technical Overview:

The Python Programming Competition aims to refine participants' skills in logical thinking and problem-solving through a set of gradually challenging Missions that take into account different age groups.

The competition focuses on training students to transform ideas into practical solutions using programming, with emphasis on:

- **Logical problem analysis.**
- **Designing simple and advanced algorithms.**
- **Implementing code in a clear and organized way that makes it easy to read and develop further.**

The competition is distinguished by the fact that it does not only test knowledge but also enhances creativity, accuracy, and discipline in coding, making it a comprehensive educational and competitive experience suitable for both beginners and advanced learners.

2. Team Composition:

- **Number of team members: 2 to 4 Students, guided by a Coach**
- **Age Group:**
  1. **Junior Category:**
     - ❖ **Ages: 10 to 12 years.**
  2. **Senior Category:**
     - ❖ **Ages: 13 to 17 years.**
  3. **Adult Category:**
     - ❖ **Ages: 18 years and above.**

3. General Rules:

- **Each participant must bring their own laptop.**
- **The code/task must be delivered before the specified time (countdown).**
- **It is strictly forbidden to use the internet during the time of the competition.**
- **After each round, the team presents their work to the judging panel.**
- **The judging panel reviews the submitted code manually and evaluates it according to specific criteria.**
- **Before the start of each round, 15 minutes will be set aside to explain the challenge and answer the teams' questions.**
- **Any contact with people outside the team during the competition time is strictly prohibited.**
- **Any intervention or assistance from coaches/supervisors during the time of the rounds will result in a first warning, and its repetition may result in the team being disqualified from the competition.**

4. Rules for Python Developers:

- **Age group allowed to participate: 10 to 18+ years old.**
- **Any code editor (IDE) such as: VS Code is allowed.**
- **The competition consists of 3 rounds, and each round is a different challenge explained by the moderator during the 15 minutes allotted before the start.**
- **Duration of each round: one hour only.**
- **The challenges are broken down by age group.**
- **Each mission must be completed on time.**
  - ❖ **The code should be:**
  - ❖ **Clean Code.**
  - ❖ **Comments illustrate the basic idea.**
  - ❖ **Properly Structured Using .(Functions/Classes)**
  - ❖ **Readable with clear variable and function names.**
- **Any output must be coordinated and professional.**
- **The use of the internet during the tour is not allowed to prevent cheating.**

5. Technical Requirements:

- **Junior Level:**
  - ❖ **Familiarity with the basics: variables, calculations, conditions, loops.**
  - ❖ **Ability to use text input /output and menus.**
  - ❖ **Write simple functions that take inputs and return outputs.**
  - ❖ **Knowledge of dictionaries and collections.**
  - ❖ **Understand simple search and sorting algorithms.**
  - ❖ **A limited set of built-in functions such as :**
    - **range(), upper(), lower(), int(), str().**
- **Senior Level:**
  - ❖ **Familiarity with all of the above at the junior level.**
  - ❖ **Handle files (open, read, write).**
  - ❖ **Apply the principles of OOP (classes, objects, simple inheritance).**
  - ❖ **Write advanced functions (Recursive, Lambda).**
  - ❖ **Knowledge of search algorithms (Binary Search) and sorting (Selection sort , insertion sort).**
  - ❖ **Familiarity with additional data structures: Stacks, Queues, Tuples.**
  - ❖ **Allows the use of additional functions such as :**
    - **max(), min(), sum(),map(), filter().**
- **Adult Level:**
  - ❖ **Familiarity with all senior level requirements.**
  - ❖ **Expansion into OOP (including multiple inheritance, polymorphism).**
  - ❖ **Exception Handling.**
  - ❖ **Create and use Modules and Packages.**
  - ❖ **Knowledge of advanced data structures: Linked List, Graph and Heap.**
  - ❖ **Knowledge of search algorithms (DFS, BFS) and advanced sorting (Merge, Sort, Quick Sort).**
  - ❖ **Allows the use of additional functions such as:**
    - **any(), all(), round(), reversed(), type(), isinstance(), hasattr().**

6. Guiding Examples :

- **Junior Level:**
- **Task 1 String Reverser:**
  Write a program that asks the user to enter a word, then reverse it and print the result.
  Expected Output:

```
Enter a word: hello
Olleh
```

- **Task 2 Even Numbers Finder:**
  Write a program that asks the user to enter a number, then prints all even numbers from 1 to that number.
  Expected Output:

```
Enter a number: 10
  2 4 6 8 10
```

- **Senior Level:**
- **Task 1 Word Counter:**
  Write a program that reads the content of a text file and then displays the number of words in it.
  Expected Output:

```
File Content: "Python is great and Python is powerful”
  Word Count = 7
```

- **Task 2 Queue Simulation:**
  Write a program that simulates a queue. Queue) Add multiple items (enqueue), then remove two items (dequeue), and view the menu after each operation.
  Expected Output:

```
Enqueue: A, B, C, D
Queue = [A, B, C, D]
  Dequeue → A
  Queue = [B, C, D]
  Dequeue → B
 Queue = [C, D]
```

- **Adult Level**:
- **Task: Coin Change Problem (Dynamic Programming)**:
  Write a program that calculates the minimum number of currencies needed to create a specific amount using certain denominations of currencies.
  Expected Output:
  Coins[5 ,2 ,1]
  Amount: 11
  Minimum coins needed = 3

7. For more Examples :

Click on this link:

https://drive.google.com/drive/folders/1YTdiyGHJS-enQ_c9UAwV2sSYQZn7DGQX?usp=drive_link